Introduction to Artificial Intelligence

DANIEL RACOCEANU
PROFESOR
SORBONNE UNIVERSITY
OCTOBER 2020
DANIEL.RACOCEANU@SORBONNE-UNIVERSITE.FR

SORBONNE UNIVERSITÉ

Document confidentiel –
ne peut être reproduit ni diffusé
sans l'accord préalable
de Sorbonne Université.

1

Forêts Aléatoires
Random Forests

DANIEL RACOCEANU
PROFESOR,
SORBONNE UNIVERSITY
OCTOBER 2020
DANIEL.RACOCEANU@SORBONNE-UNIVERSITE.FR

SORBONNE UNIVERSITÉ

Document confidentiel –
ne peut être reproduit ni diffusé
sans l'accord préalable
de Sorbonne Université.

3

About 25000 citations … !

Leo Breiman

- ✓ 1928 – 2005
- ✓ Statistics Department, University of California, Berkeley

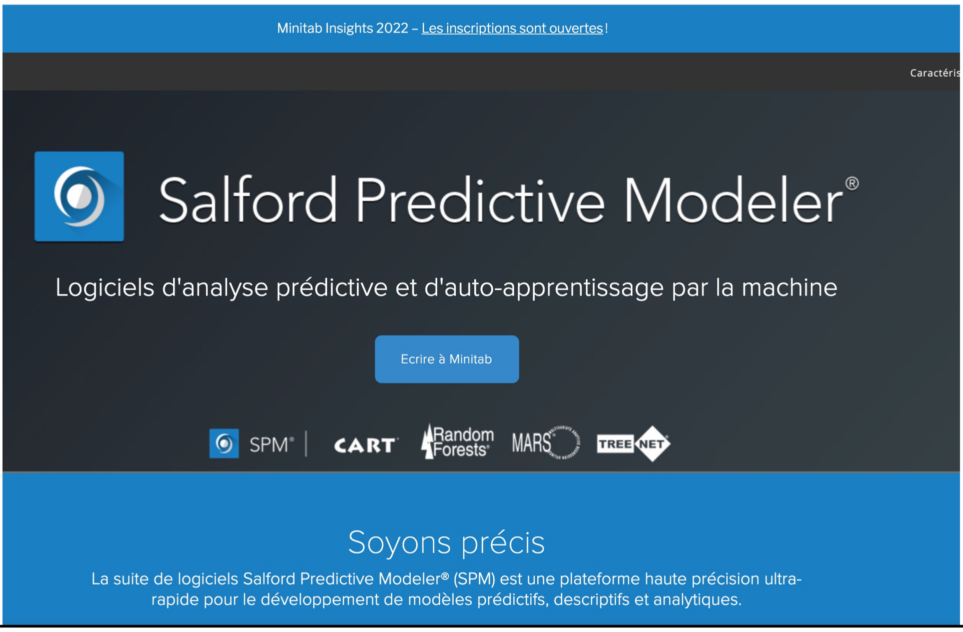
Machine Learning archive, Volume 45 Issue 1, 2001

Random Forests™ is a trademark of Leo Breiman and Adele Cutler and is licensed exclusively to Salford Systems for the commercial release of the software. Their trademarks also include RF™, RandomForests™,

RandomForest™ and Random Forest™.

- ✓ Salfort Systems (San Diego) : https://www.salford-systems.com/

https://www.salford-systems.com/



Minitab Insights 2022 – Les inscriptions sont ouvertes !

Caractéris

Salford Predictive Modeler®

Logiciels d'analyse prédictive et d'auto-apprentissage par la machine

Soyons précis

La suite de logiciels Salford Predictive Modeler® (SPM) est une plateforme haute précision ultra-rapide pour le développement de modèles prédictifs, descriptifs et analytiques.

Random Forest

Definition

- ✓ Collection of un-pruned CARTs
- ✓ Rule to combine individual tree decisions

Purpose

- ✓ Improve prediction accuracy

Principle

- ✓ Encouraging diversity among the tree

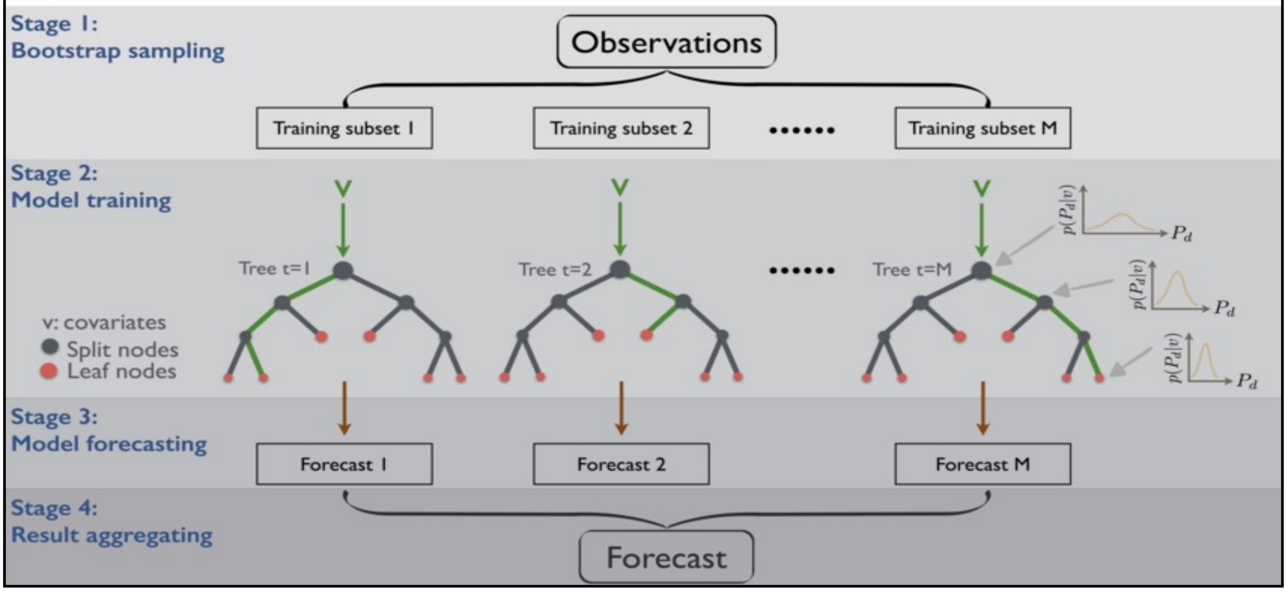
Solution: randomness

- ✓ Bagging
- ✓ Random decision trees (rCART)

6

6

Schematic of the RF algorithm based on the Bagging (Bootstrap + Aggregating) method



Stage 1:
Bootstrap sampling

Observations

Training subset 1 | Training subset 2 | •••••• | Training subset M

Stage 2:
Model training

v: covariates
● Split nodes
● Leaf nodes

Stage 3:
Model forecasting

Forecast 1 | Forecast 2 | Forecast M

Stage 4:
Result aggregating

Forecast

7

RF: Bagging …

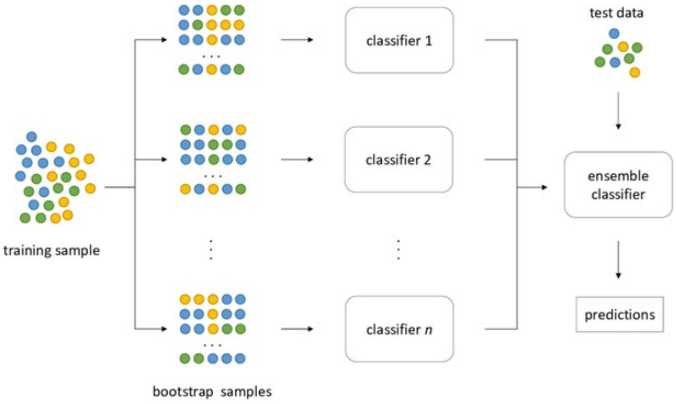
Bagging = Bootstrap aggregation

Technique of ensemble learning…

- ✓ … to avoid over-fitting
  - ➢ Important since trees are un-pruned
- ✓ … to improve stability and accuracy

Two steps

- ✓ Bootstrap sample set
- ✓ Aggregation



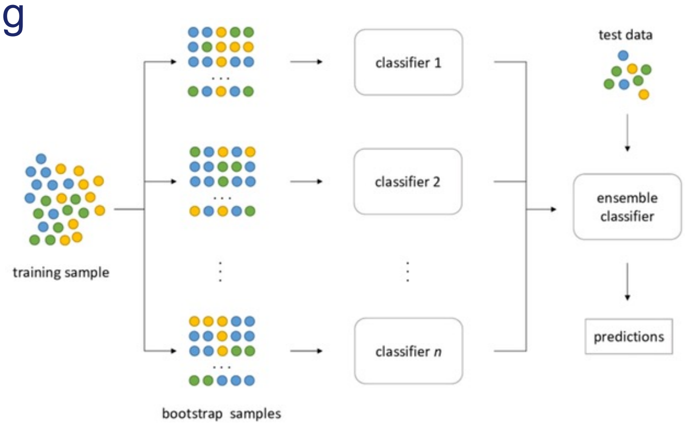
8

8

RF: Bagging … the rationale

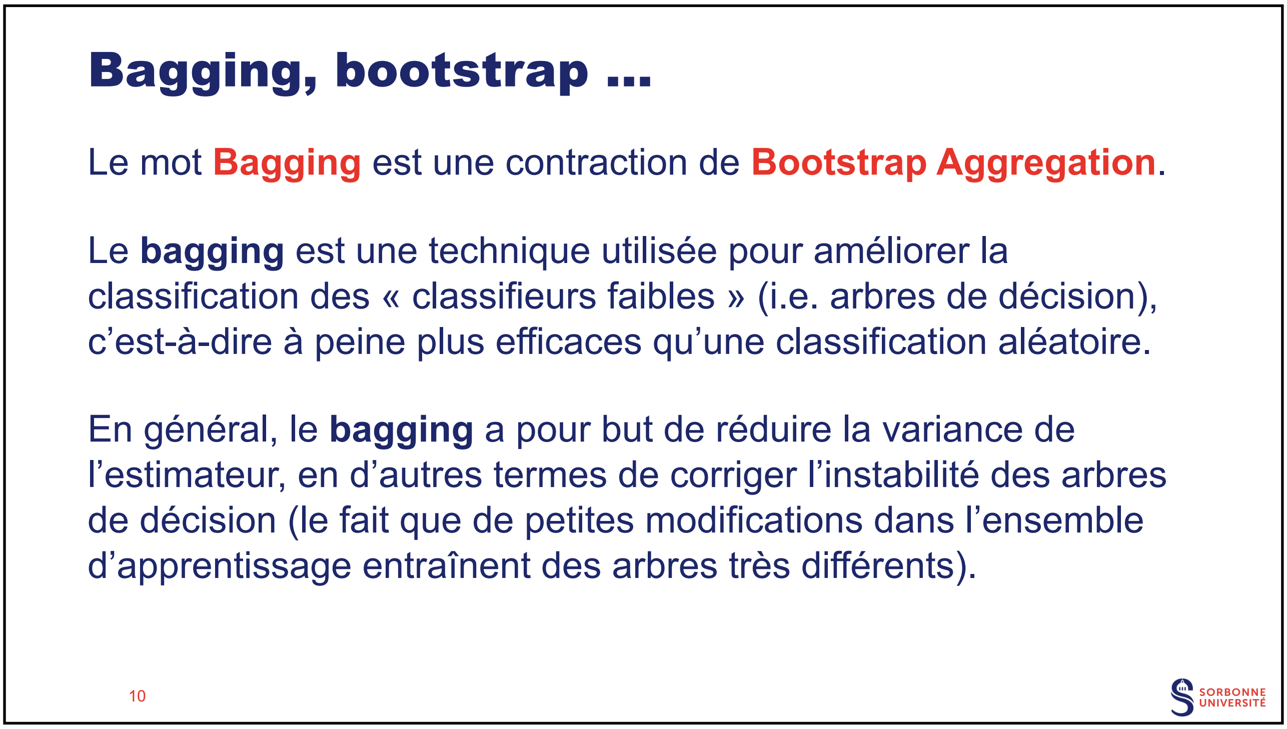
The combination of learning models increase the classification accuracy (**bagging**)

Goal of **Bagging** -> to average noisy and unbiased models to create a model with low variance



9

9

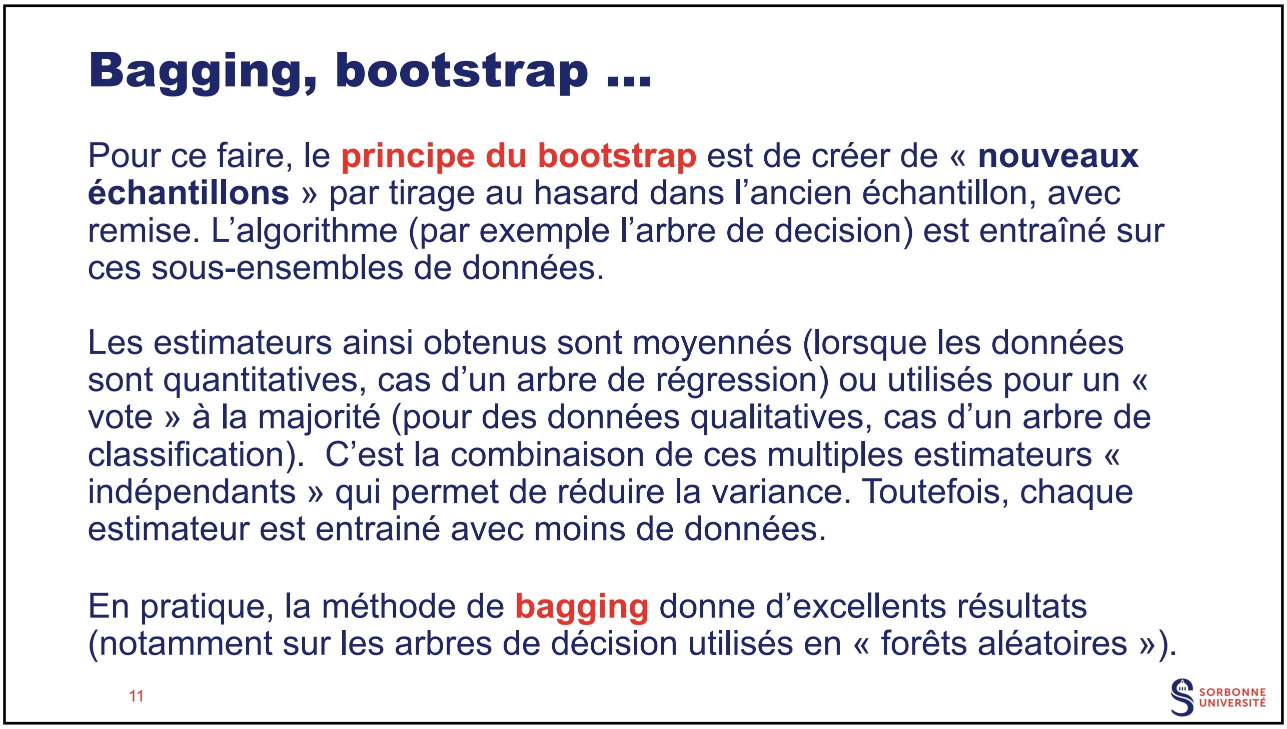


Bagging, bootstrap …

Le mot **Bagging** est une contraction de **Bootstrap Aggregation**.

Le **bagging** est une technique utilisée pour améliorer la classification des « classifieurs faibles » (i.e. arbres de décision), c'est-à-dire à peine plus efficaces qu'une classification aléatoire.

En général, le **bagging** a pour but de réduire la variance de l'estimateur, en d'autres termes de corriger l'instabilité des arbres de décision (le fait que de petites modifications dans l'ensemble d'apprentissage entraînent des arbres très différents).

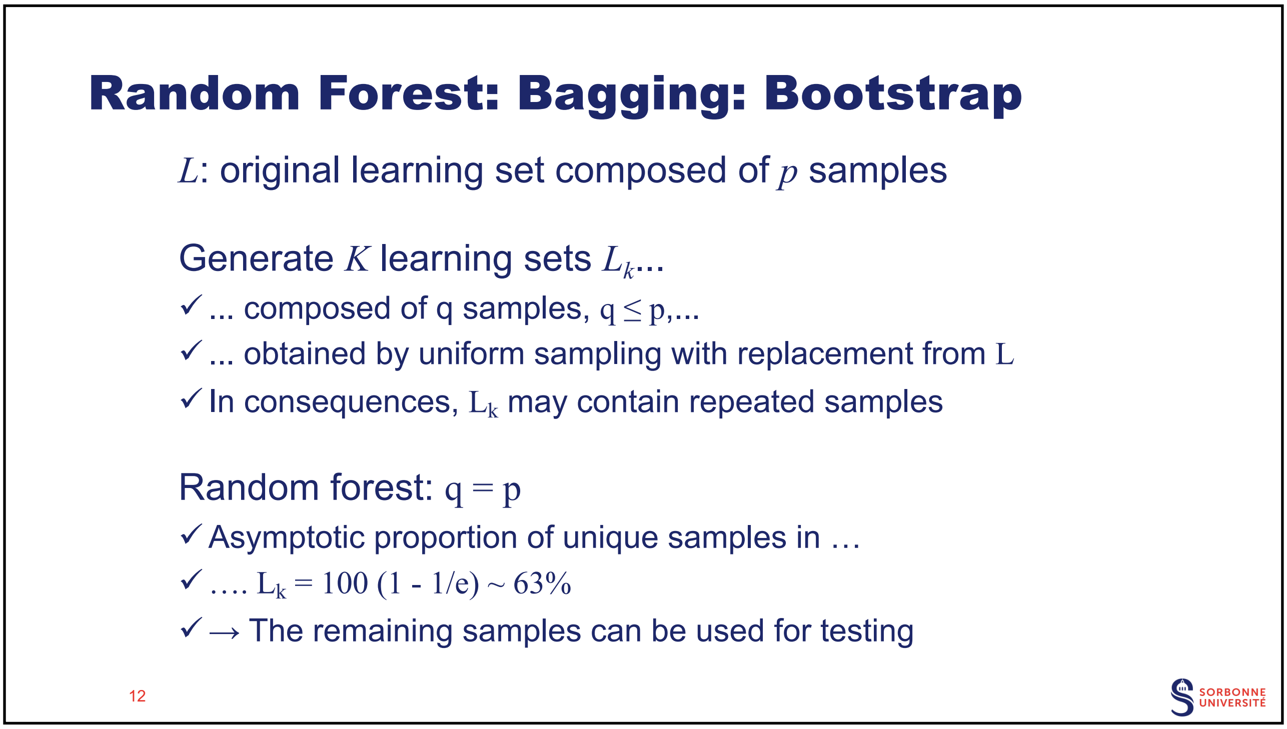


Bagging, bootstrap …

Pour ce faire, le **principe du bootstrap** est de créer de « **nouveaux échantillons** » par tirage au hasard dans l'ancien échantillon, avec remise. L'algorithme (par exemple l'arbre de decision) est entraîné sur ces sous-ensembles de données.

Les estimateurs ainsi obtenus sont moyennés (lorsque les données sont quantitatives, cas d'un arbre de régression) ou utilisés pour un « vote » à la majorité (pour des données qualitatives, cas d'un arbre de classification). C'est la combinaison de ces multiples estimateurs « indépendants » qui permet de réduire la variance. Toutefois, chaque estimateur est entrainé avec moins de données.

En pratique, la méthode de **bagging** donne d'excellents résultats (notamment sur les arbres de décision utilisés en « forêts aléatoires »).



Random Forest: Bagging: Bootstrap

L: original learning set composed of p samples

Generate K learning sets L_k...

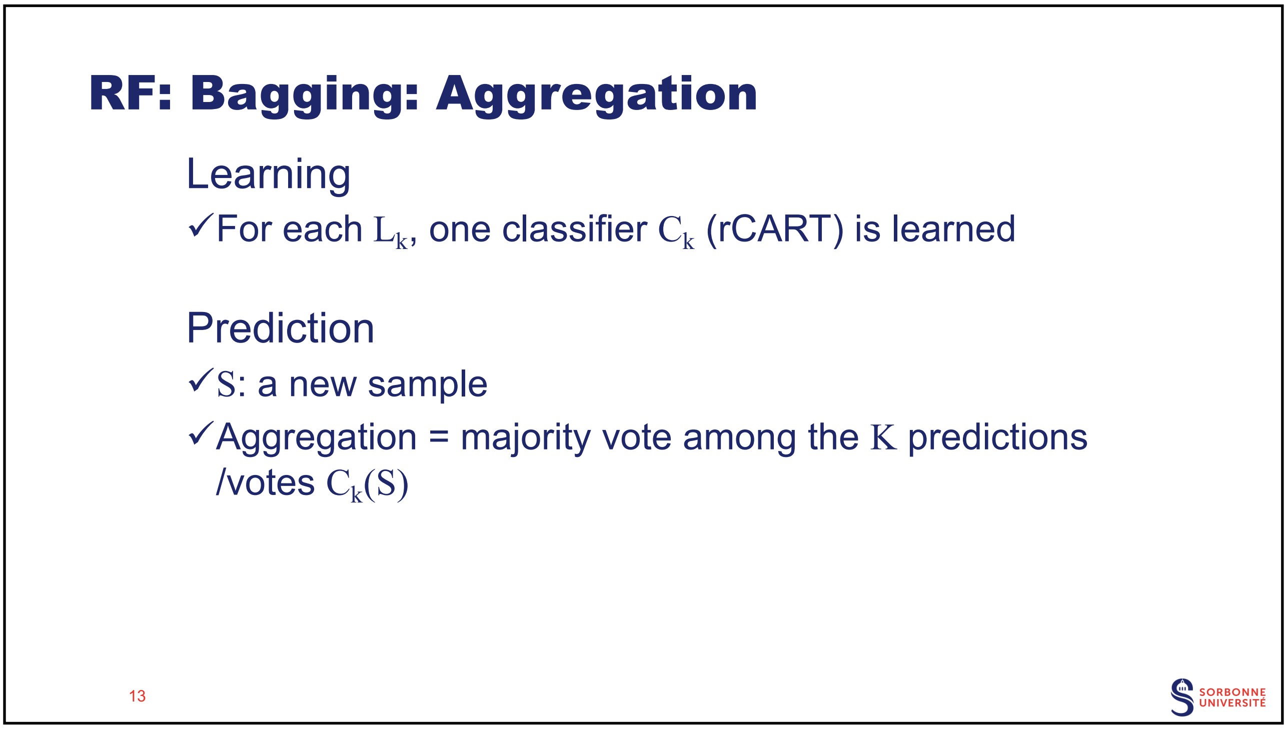
- ✓ ... composed of q samples, $q \leq p$,...
- ✓ ... obtained by uniform sampling with replacement from L
- ✓ In consequences, L_k may contain repeated samples

Random forest: $q = p$

- ✓ Asymptotic proportion of unique samples in …
- ✓ …. $L_k = 100\ (1 - 1/e) \sim 63\%$
- ✓ → The remaining samples can be used for testing

12

12



RF: Bagging: Aggregation

Learning

- ✓ For each L_k, one classifier C_k (rCART) is learned

Prediction

- ✓ S: a new sample
- ✓ Aggregation = majority vote among the K predictions /votes $C_k(S)$

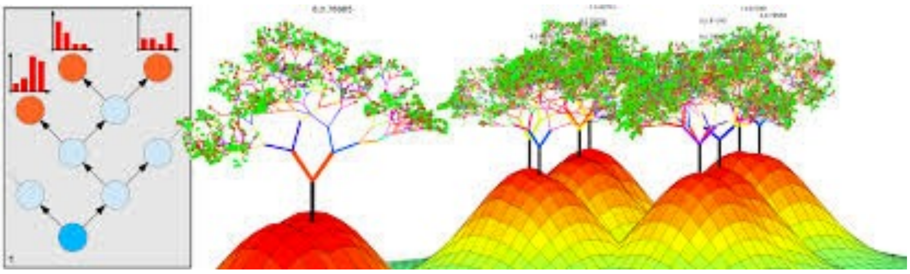
13

13

Random Forest vs. Trees

The random forest takes the decision tree to the next level by combining trees with the notion of an ensemble.

Random forest algorithm works as a large collection of decorrelated decision trees.
Thus, in ensemble terms, the **trees are weak learners** and the **random forest is a strong learner**.



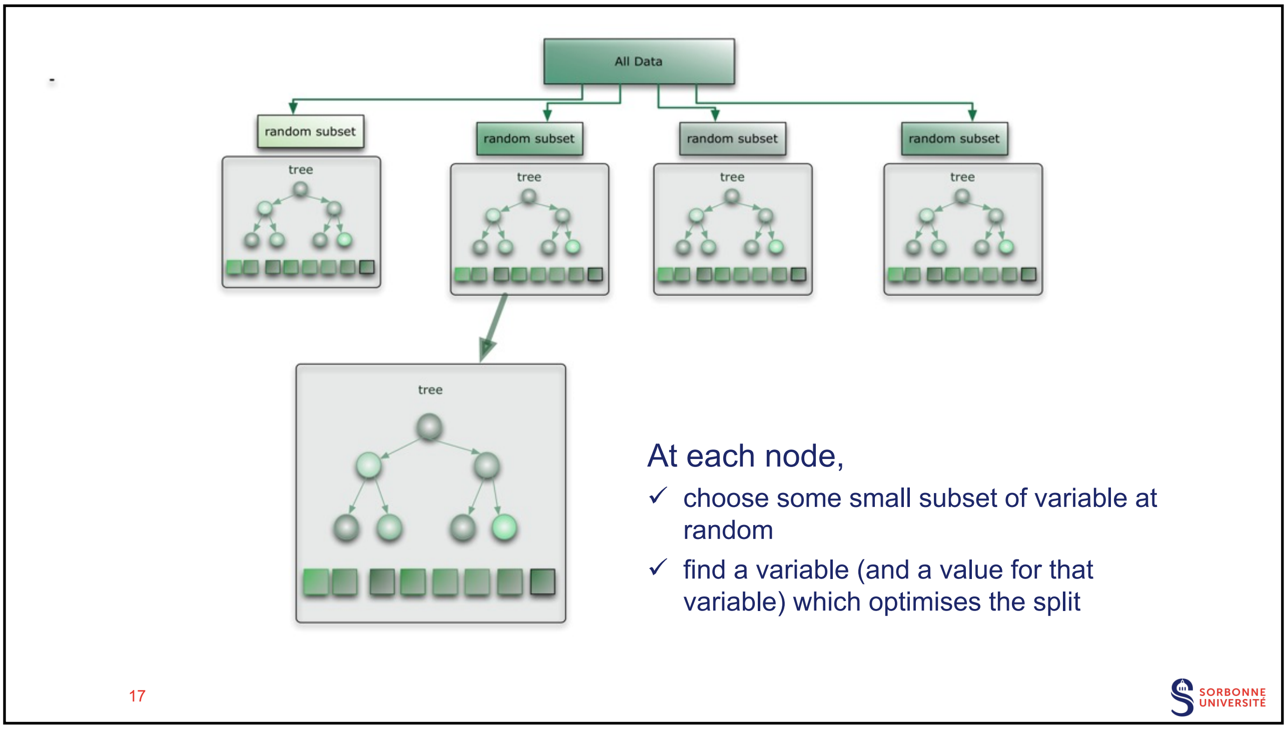
15

15

Similarité avec le brainstorming

16

16

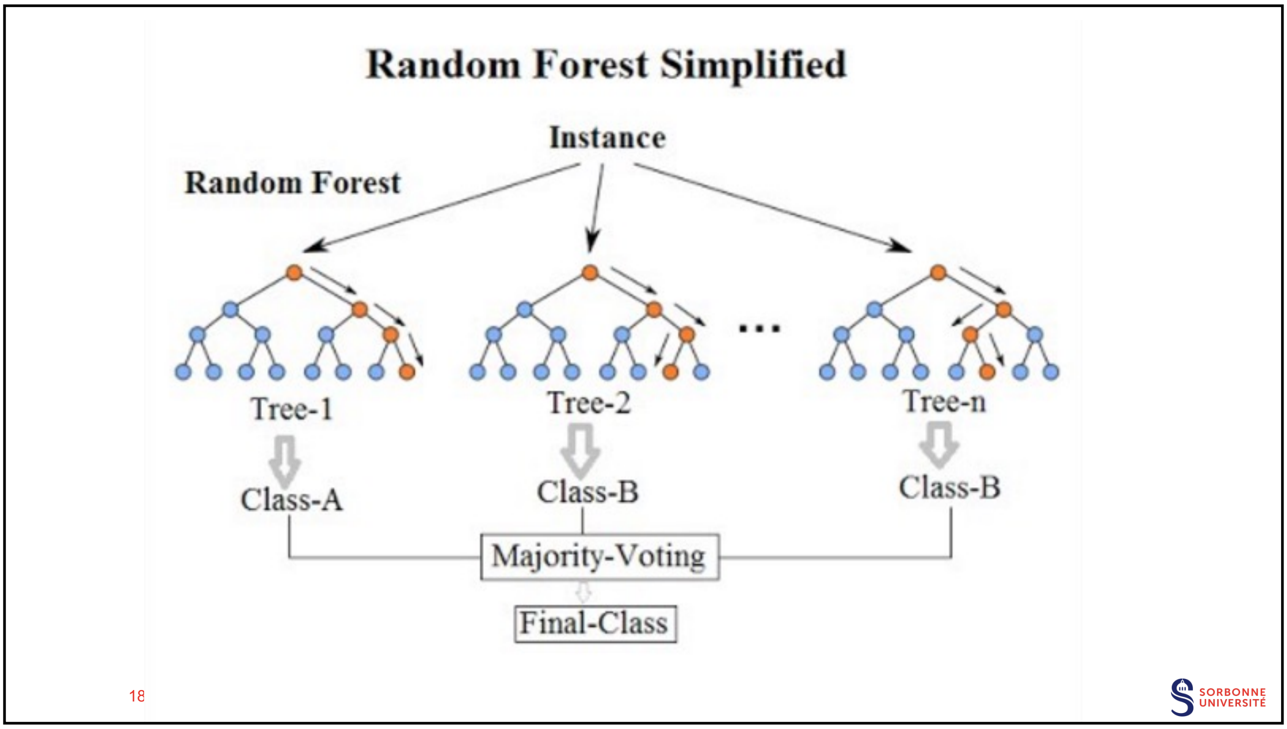


At each node,

- ✓ choose some small subset of variable at random
- ✓ find a variable (and a value for that variable) which optimises the split

17

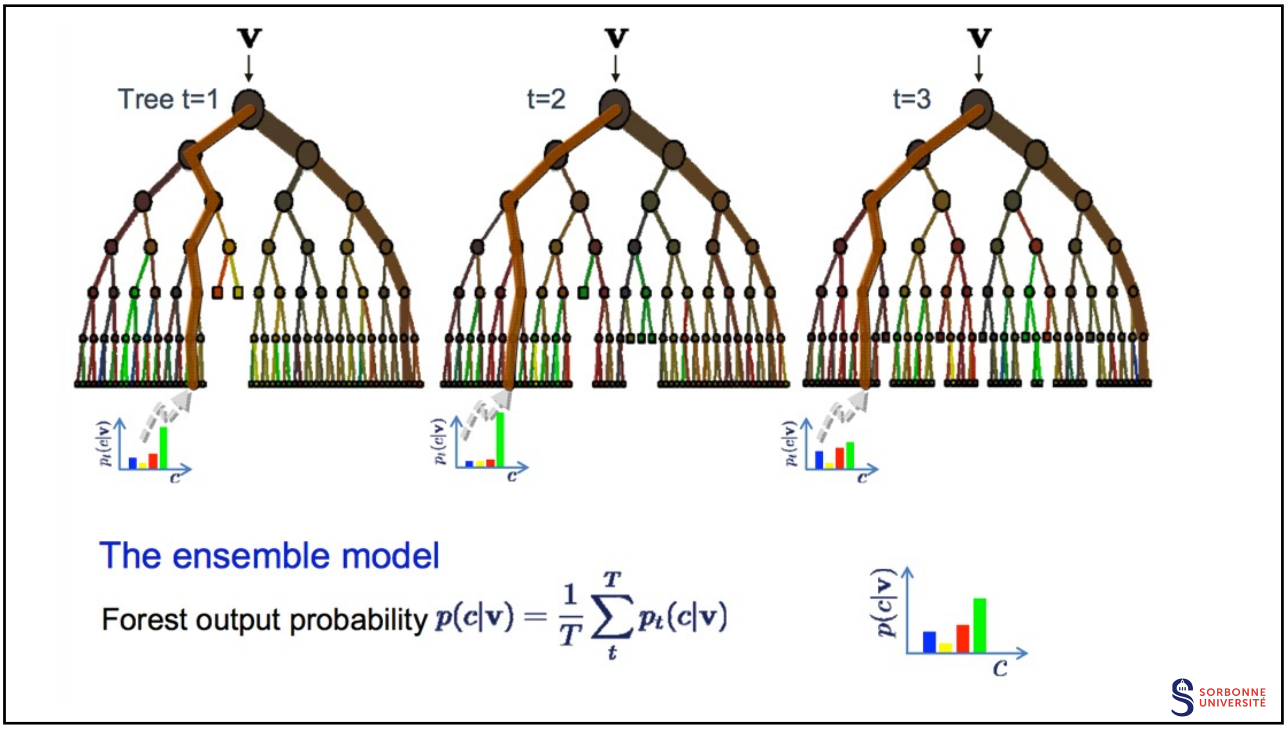
17



Random Forest Simplified

18

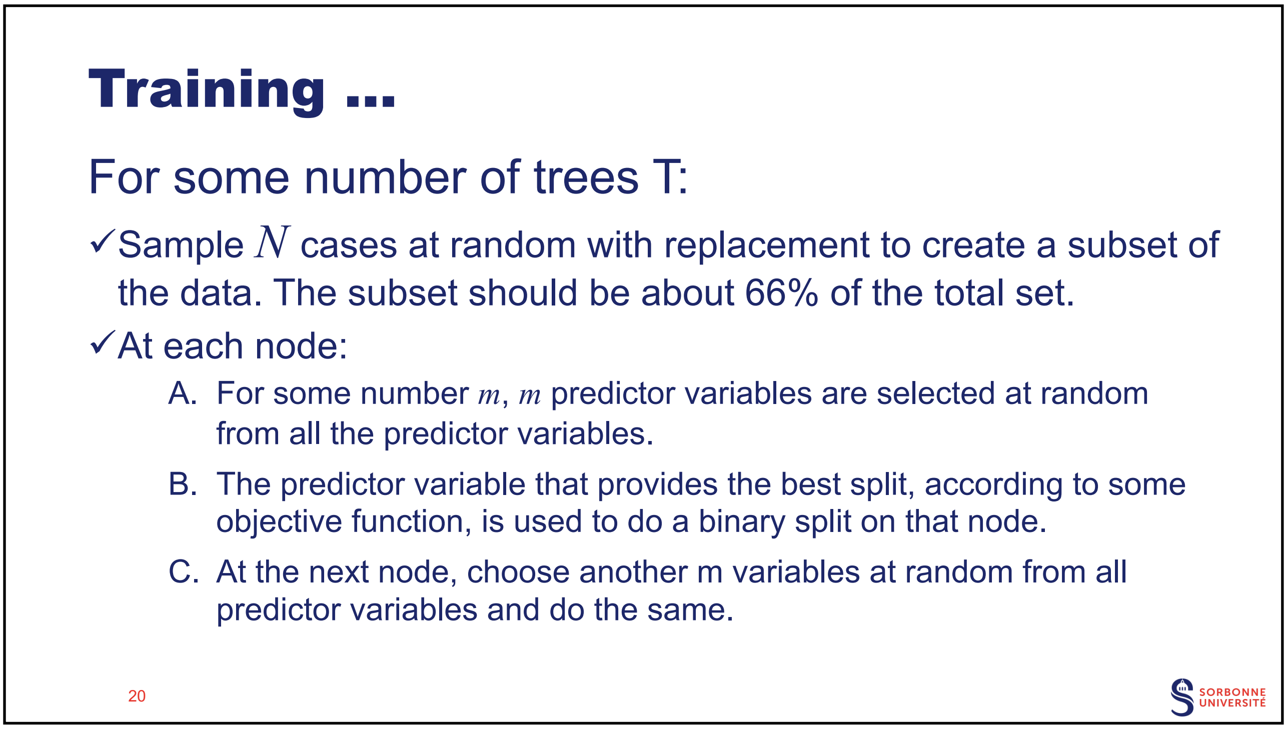
18



The ensemble model

Forest output probability $p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v})$

19



Training …

For some number of trees T:

✓Sample N cases at random with replacement to create a subset of the data. The subset should be about 66% of the total set.

✓At each node:

A. For some number m, m predictor variables are selected at random from all the predictor variables.

B. The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.

C. At the next node, choose another m variables at random from all predictor variables and do the same.

20

SORBONNE UNIVERSITÉ

20

Depending upon the value of m, there are three slightly different systems:

- ✓ Random splitter selection: $m = 1$
- ✓ Breiman’s bagger: m = total number of predictor variables
- ✓ Random forest: $m <<$ number of predictor variables. Brieman suggests three possible values for m: $\frac{1}{2}\sqrt{m}$, $\sqrt{m}$, and $2\sqrt{m}$

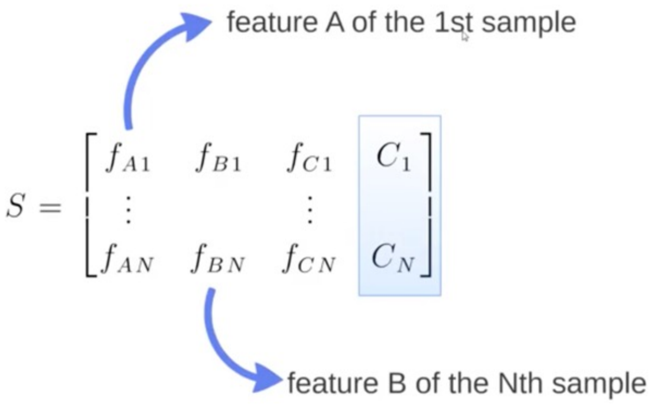
Running a Random Forest

When a new input is entered into the system, it is run down all of the trees. The result may either be an average or weighted average of all of the terminal nodes that are reached, or, in the case of categorical variables, a voting majority.

Note that:

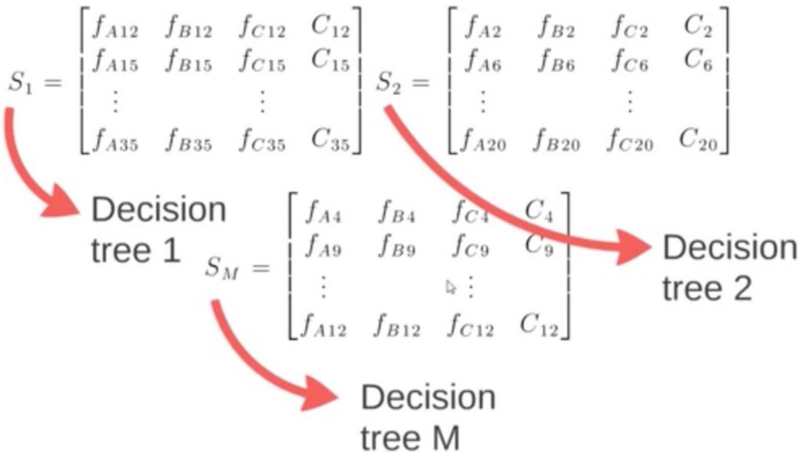
- ✓ With a large number of predictors, the eligible predictor set will be quite different from node to node.
- ✓ The greater the inter-tree correlation, the greater the random forest error rate, so one pressure on the model is to have the trees as uncorrelated as possible.
- ✓ As m goes down, both inter-tree correlation and the strength of individual trees go down. So some optimal value of m must be discovered.

Training Features ... exemple



$$S = \begin{bmatrix} f_{A1} & f_{B1} & f_{C1} & C_1 \\ \vdots & & \vdots & \\ f_{AN} & f_{BN} & f_{CN} & C_N \end{bmatrix}$$

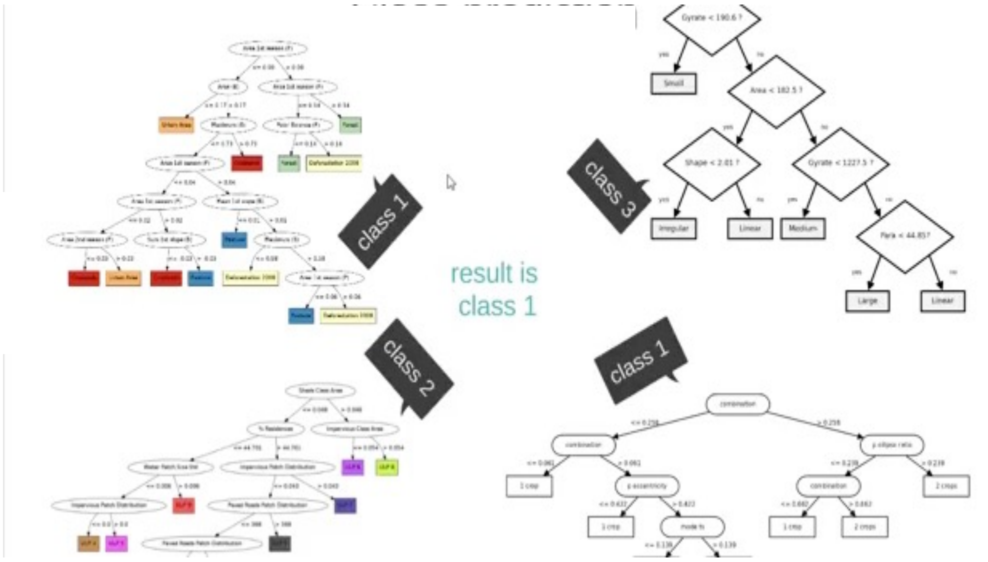
Create Random Subsets



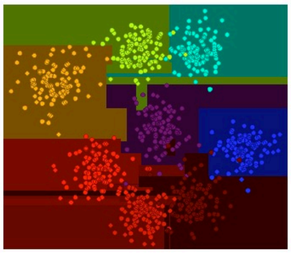
$$S_1 = \begin{bmatrix} f_{A12} & f_{B12} & f_{C12} & C_{12} \\ f_{A15} & f_{B15} & f_{C15} & C_{15} \\ \vdots & & \vdots & \\ f_{A35} & f_{B35} & f_{C35} & C_{35} \end{bmatrix} \quad S_2 = \begin{bmatrix} f_{A2} & f_{B2} & f_{C2} & C_2 \\ f_{A6} & f_{B6} & f_{C6} & C_6 \\ \vdots & & \vdots & \\ f_{A20} & f_{B20} & f_{C20} & C_{20} \end{bmatrix}$$

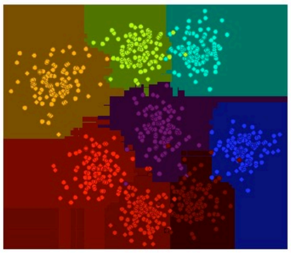
$$S_M = \begin{bmatrix} f_{A4} & f_{B4} & f_{C4} & C_4 \\ f_{A9} & f_{B9} & f_{C9} & C_9 \\ \vdots & & \vdots & \\ f_{A12} & f_{B12} & f_{C12} & C_{12} \end{bmatrix}$$

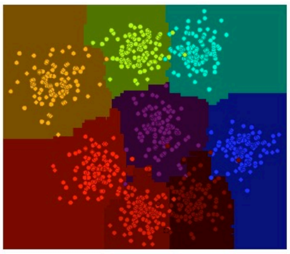
Class Prediction

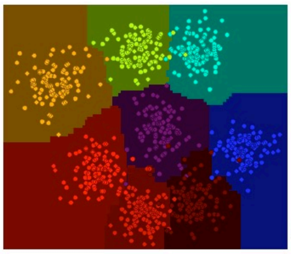


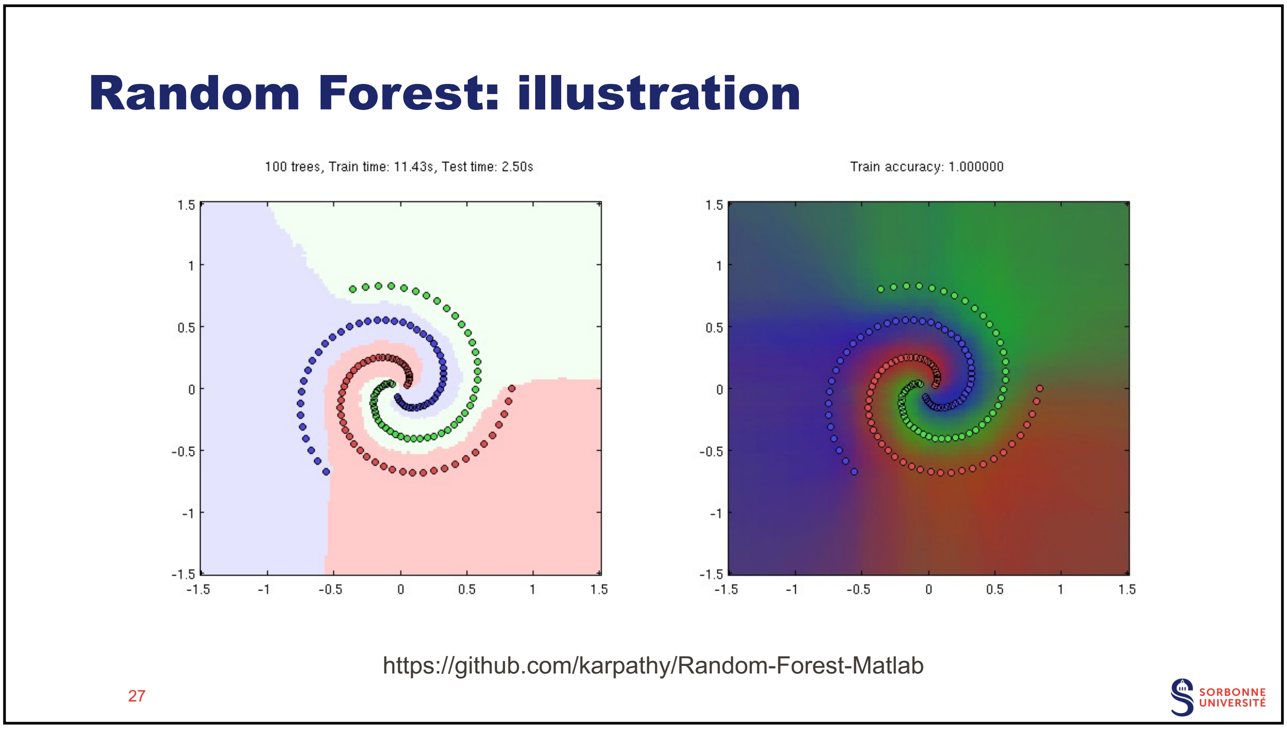
Random Forest: illustration


1 rCART


10 rCARTs

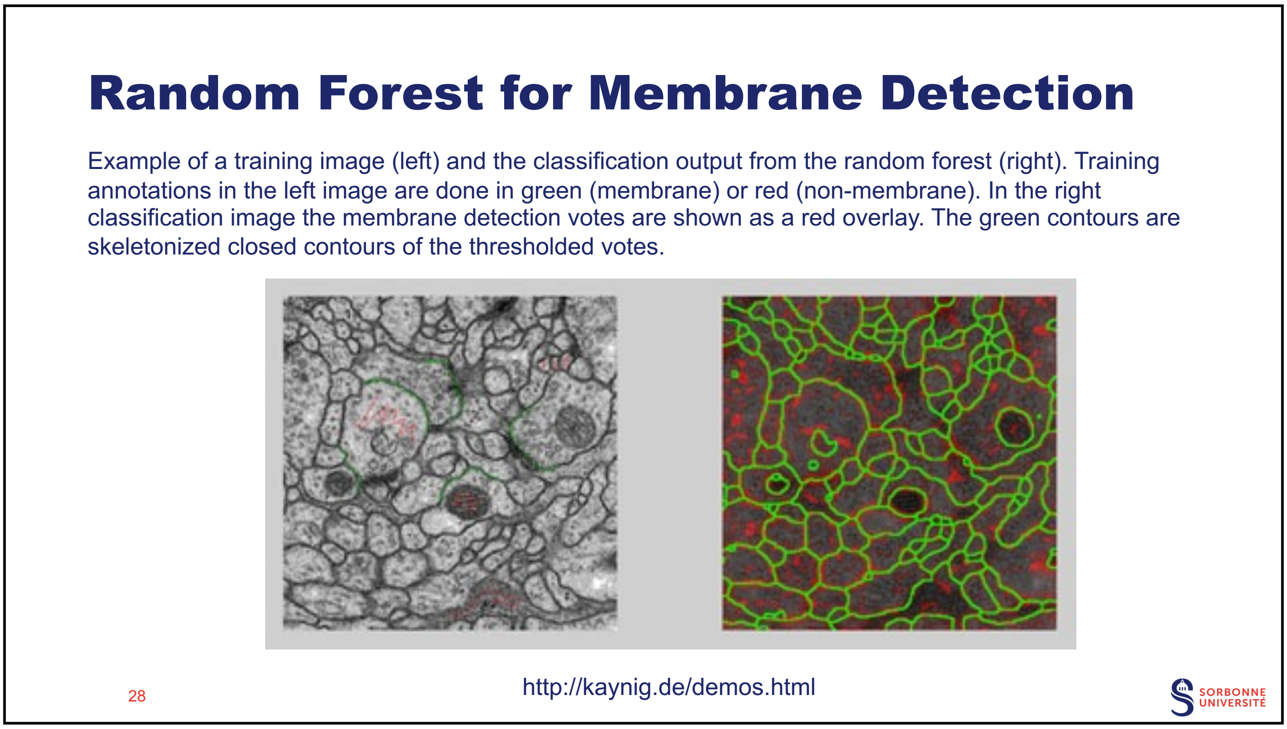

100 rCARTs


500 rCARTs



Random Forest: illustration

https://github.com/karpathy/Random-Forest-Matlab



Random Forest for Membrane Detection

Example of a training image (left) and the classification output from the random forest (right). Training annotations in the left image are done in green (membrane) or red (non-membrane). In the right classification image the membrane detection votes are shown as a red overlay. The green contours are skeletonized closed contours of the thresholded votes.

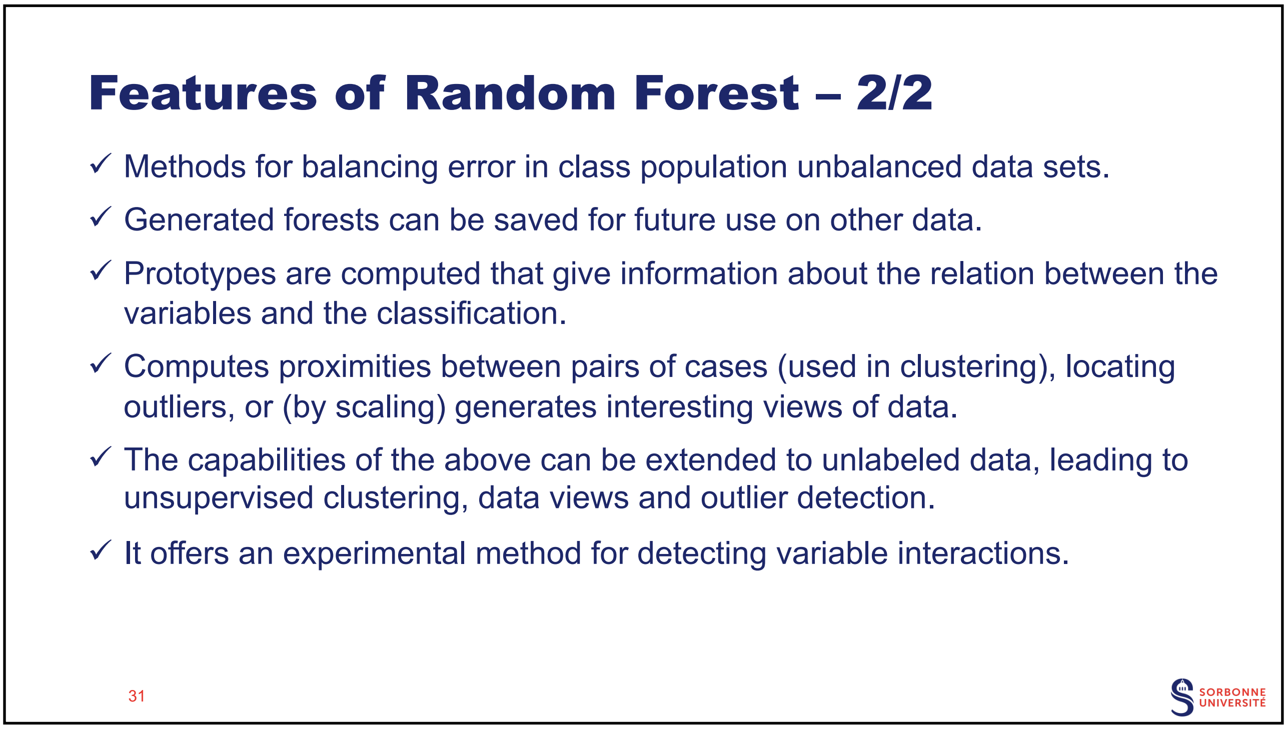
http://kaynig.de/demos.html

Properties of Random Forest

	RF	CART	kNN	SVM
Intrinsically multiclass	🟢	🟢	🟢	🟠
Handles Apple and Orange features	🟢	🟢	🔴	🔴
Robustness to outliers	🟢	🟢	🟢	🟠
Works w/ "small" learning set	🔴	🔴	🔴	🟢
Scalability (large learning set)	🟢	🟢	🔴	🔴
Prediction accuracy	🟢	🔴	🟠	🟢
Parameter tuning	🟢	🟢	🟠	🔴

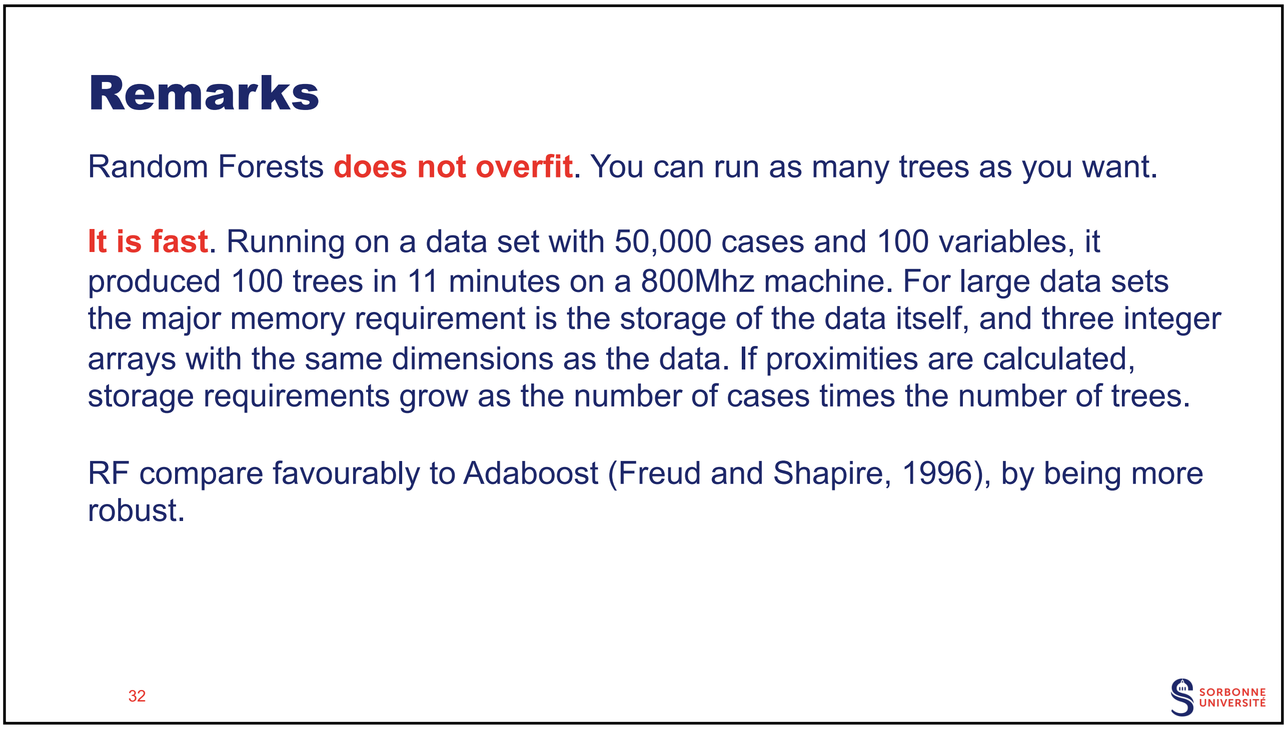
Features of Random Forest -1/2

- ✓ It is unexcelled in accuracy among current algorithms.
- ✓ It runs efficiently on large data bases.
  - ➢ It can handle thousands of input variables without variable deletion.
- ✓ It gives estimates of what variables are important in the classification.
- ✓ It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- ✓ It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.



Features of Random Forest – 2/2

- ✓ Methods for balancing error in class population unbalanced data sets.
- ✓ Generated forests can be saved for future use on other data.
- ✓ Prototypes are computed that give information about the relation between the variables and the classification.
- ✓ Computes proximities between pairs of cases (used in clustering), locating outliers, or (by scaling) generates interesting views of data.
- ✓ The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- ✓ It offers an experimental method for detecting variable interactions.



Remarks

Random Forests **does not overfit**. You can run as many trees as you want.

It is fast. Running on a data set with 50,000 cases and 100 variables, it produced 100 trees in 11 minutes on a 800Mhz machine. For large data sets the major memory requirement is the storage of the data itself, and three integer arrays with the same dimensions as the data. If proximities are calculated, storage requirements grow as the number of cases times the number of trees.

RF compare favourably to Adaboost (Freud and Shapire, 1996), by being more robust.

Strengths and weaknesses

Random forest runtimes are **quite fast**, and they are able to deal with **unbalanced and missing data**.

Random Forest weaknesses are that when used for regression they cannot predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy. Of course, the best test of any algorithm is how well it works upon your own data set.

A known drawback of Random Forest is it can be a **black box approach to machine learning**, where it's difficult to get insights into each feature's importance, and to go through each tree to understand how it came up with its prediction.

Limitations

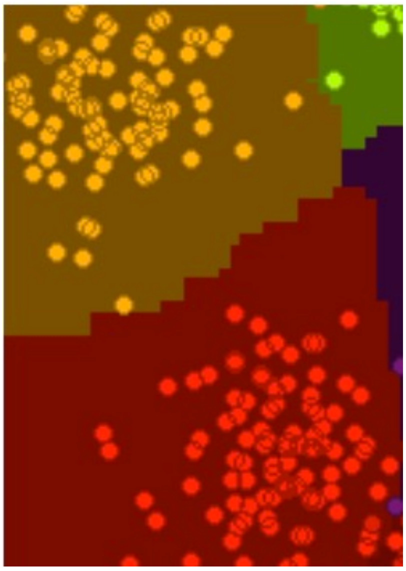
Oblique/curved frontiers

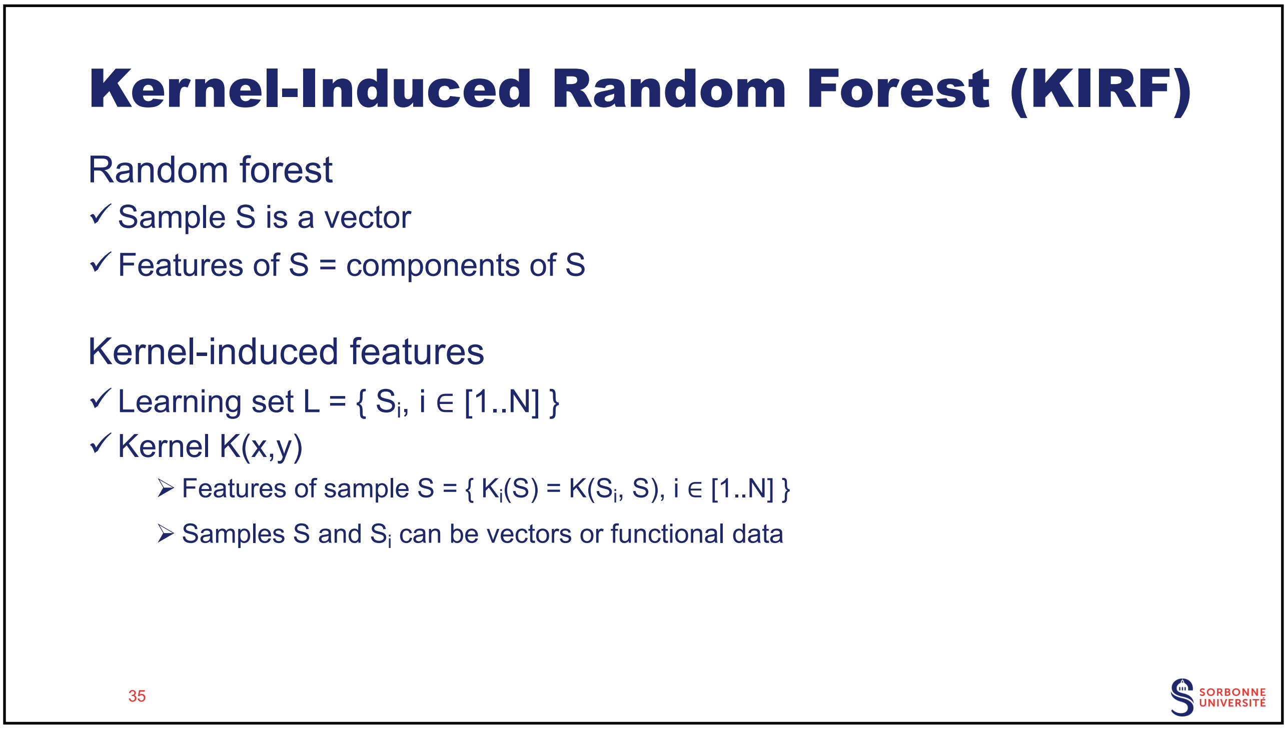
✓ Staircase effect

✓ Many pieces of hyperplanes

Fundamentally discrete

✓ Functional data? (Example: curves)





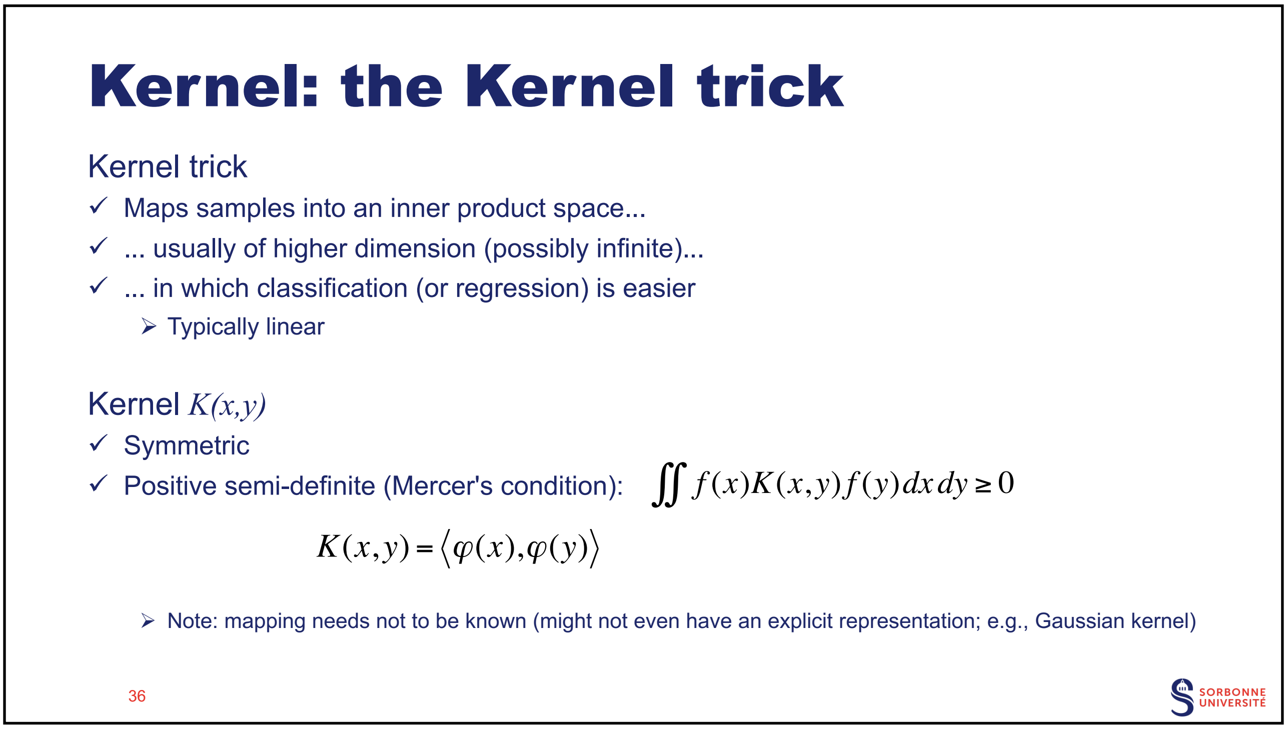
Kernel-Induced Random Forest (KIRF)

Random forest

- ✓ Sample S is a vector
- ✓ Features of S = components of S

Kernel-induced features

- ✓ Learning set L = { S_i, i ∈ [1..N] }
- ✓ Kernel K(x,y)
  - ➢ Features of sample S = { $K_i(S) = K(S_i, S)$, i ∈ [1..N] }
  - ➢ Samples S and S_i can be vectors or functional data



Kernel: the Kernel trick

Kernel trick

- ✓ Maps samples into an inner product space…
- ✓ … usually of higher dimension (possibly infinite)…
- ✓ … in which classification (or regression) is easier
  - ➢ Typically linear

Kernel *K(x,y)*

- ✓ Symmetric
- ✓ Positive semi-definite (Mercer's condition): $\iint f(x)K(x,y)f(y)\,dx\,dy \geq 0$

$$K(x,y) = \langle \varphi(x), \varphi(y) \rangle$$

- ➢ Note: mapping needs not to be known (might not even have an explicit representation; e.g., Gaussian kernel)

Kernel: Examples

Polynomial (homogeneous): $K(x,y) = (x \cdot y)^d$

Polynomial (inhomogeneous): $K(x,y) = (x \cdot y + 1)^d$

Hyperbolic tangent: $K(x,y) = \tanh(\alpha x \cdot y + \beta)$

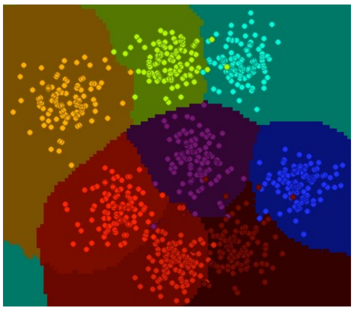
Gaussian: $K(x,y) = \exp(-\gamma |x - y|^2)$

- ✓ Function of the distance between samples
- ✓ Straightforward application to functional data of a metric space
  - ➢ E.g., curves

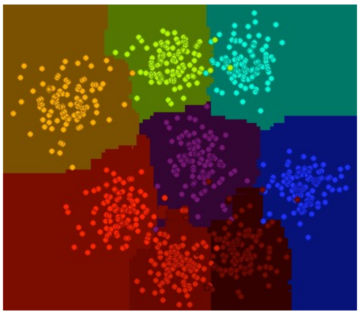
KIRF: Illustration

Gaussian kernel

- ✓ Some similarity with vantage-point tree



KIRF w/ 100 rCARTs



Reminder: RF w/ 100 rCARTs

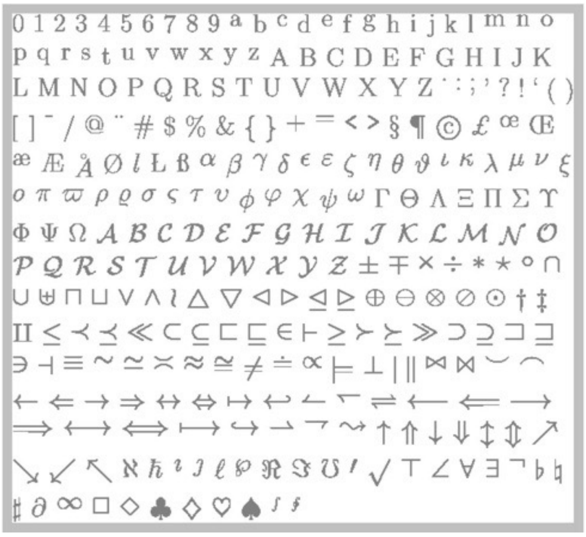
KIRF: Limitations

- Which kernel?
  - ✓ Which kernel parameters?
- No “orange and apple” handling anymore
  - ✓ $(x \cdot y$ or $(x - y)^2)$
- Computational load (kernel evaluations)
  - ✓ Especially during learning
- Needs to store samples
  - ✓ (Instead of feature indices in Random forest)

SORBONNE UNIVERSITÉ

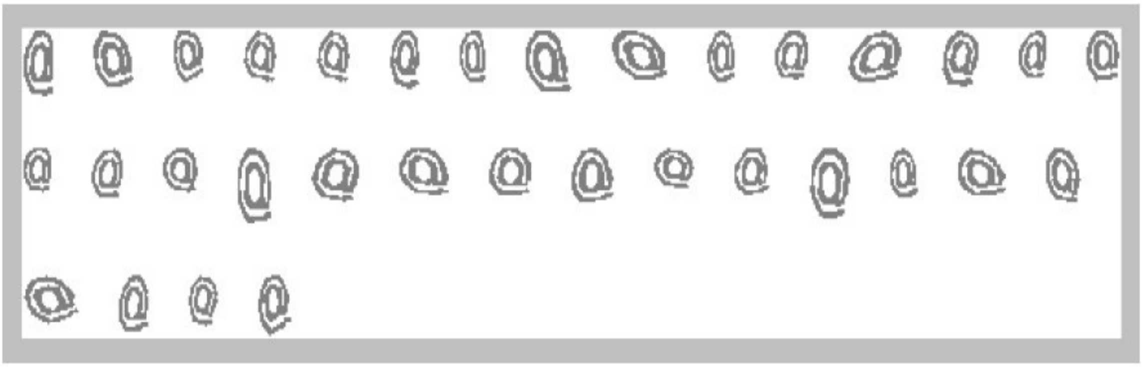
SHAPE QUANTIZATION:

RECOGNIZING LATEX SYMBOLS



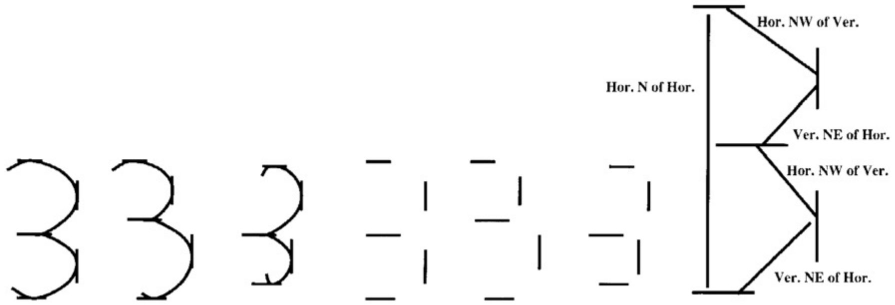
SORBONNE UNIVERSITÉ

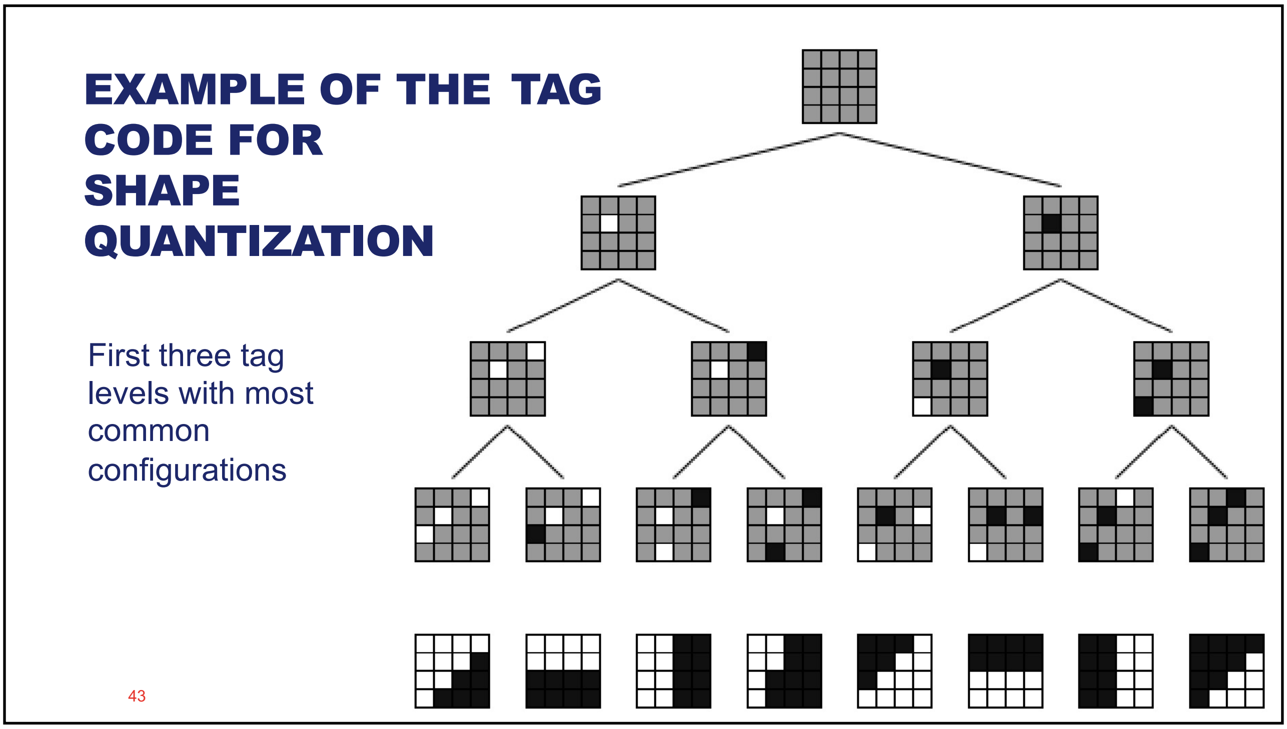
GENERATING TRAINING DATA BY PERTURBATION



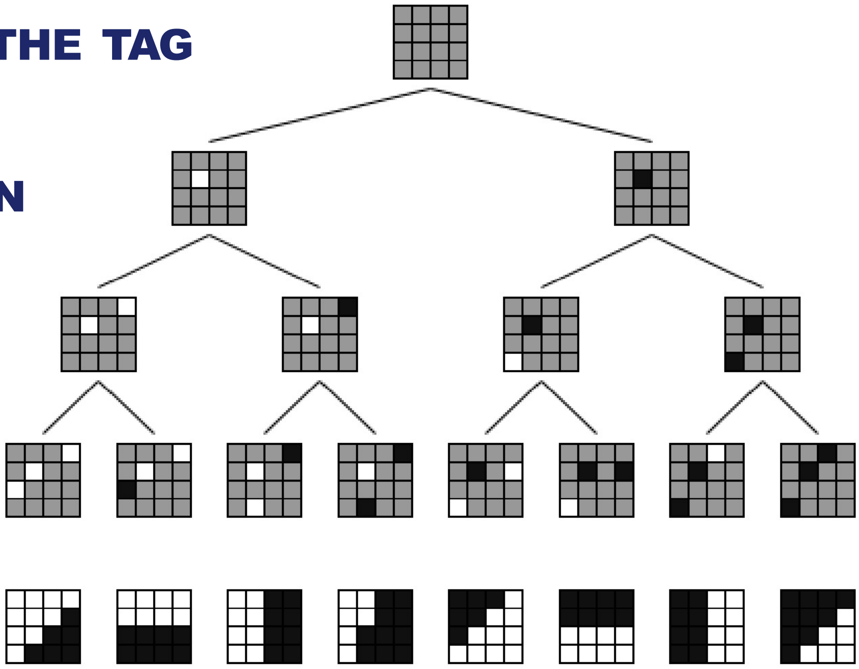
SHAPE QUERIES

(Left) Three curves corresponding to the digit "3." (Middle) Three tangent configurations determining these shapes via spline interpolation. (Right) Graphical description of relations between locations of derivatives consistent with all three configurations





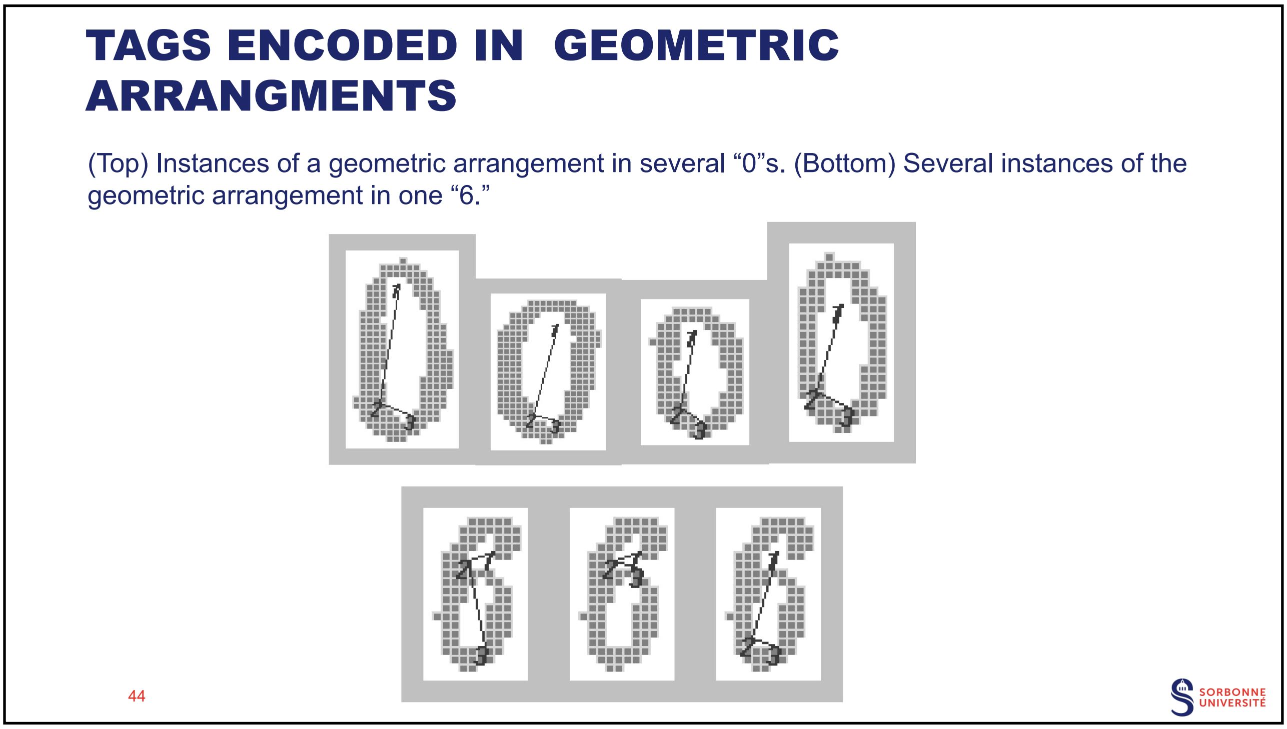
EXAMPLE OF THE TAG CODE FOR SHAPE QUANTIZATION



First three tag levels with most common configurations

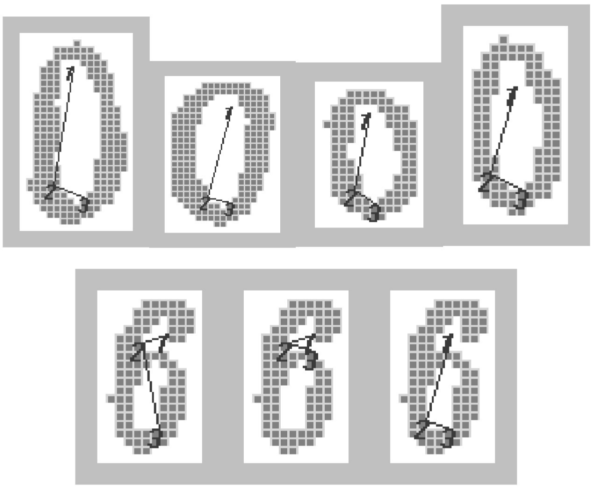
43

43



TAGS ENCODED IN GEOMETRIC ARRANGMENTS

(Top) Instances of a geometric arrangement in several “0”s. (Bottom) Several instances of the geometric arrangement in one “6.”

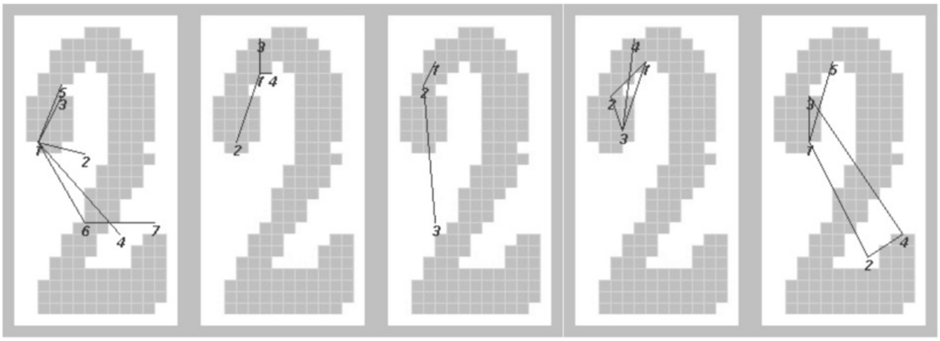


44

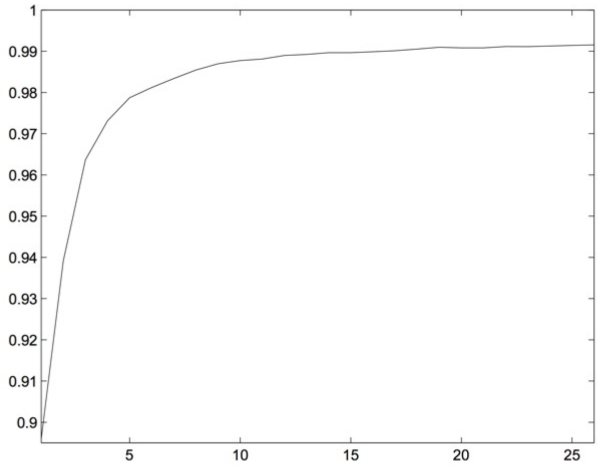
44

EXAMPLE STRUCTURE GRAPHS FROM MULTIPLE RANDOMIZED TREES

Graphs found in an image at terminal nodes of five different trees.

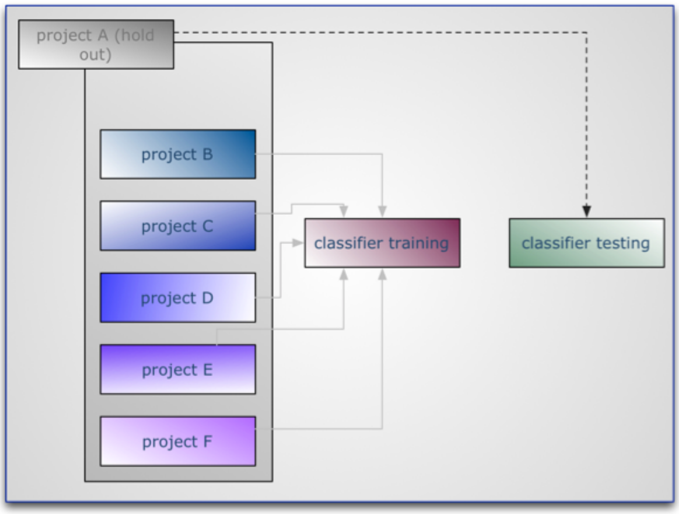


CLASSIFICATION RATE



- Classification rate versus number of trees.

Leave One Out (LOO)



project A (hold out)

project B

project C

project D

project E

project F

classifier training

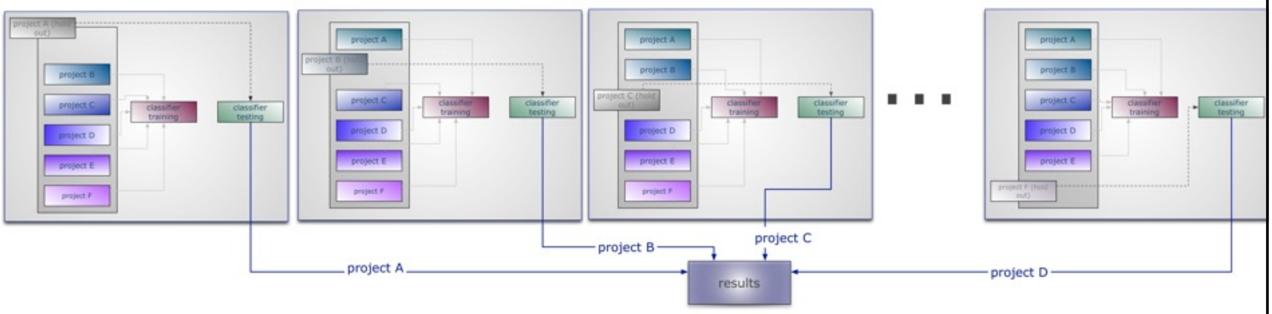
classifier testing

47

47

Cross Validation

- We have projects A through F. We train a classifier on projects A through E, and test on F. Then we train on A and C through F, and test on B. We do this in turn, holding out each project, until we train on A through E and test on F. Each project is a subject in our experiment, with subjects A through F.



project A

project B

project C

project D

results

Finally, we collect the results from each cross-validation run for statistical analysis.

48